"Express Mail" mailing label number:

EV 335894996 US

ARITHMETIC PROCESSOR UTILIZING MULTI-TABLE LOOK UP TO OBTAIN RECIPROCAL OPERANDS

Willard S. Briggs
David W. Matula

BACKGROUND

Field of the Disclosure

[0001] The present disclosure generally relates to computer systems, and more particularly to a computer system providing results of arithmetic operations.

Description of the Related Art

[0002] A computer processor performs arithmetic operations on different types of numbers, or operands. For example, the simplest operations involve integer operands, which are represented using a "fixed-point" notation. Non-integers are typically represented according to a "floating-point" notation.

[0003] Many processors handle floating-point operations within a floating-point unit (FPU). Floating-point processing typically includes addition, multiplication and division operations, may also include other special mathematical operations on a single operand, such as the square root ($\sqrt{x}$), reciprocal square root ($1/\sqrt{x}$), and reciprocal (1/x) representing functions.

[0004] Floating point units (and other arithmetic processors) commonly use multiplier based algorithms for division. These division algorithms initially employ a seed reciprocal of the divisor provided by a lookup table system..

[0005] The seed reciprocals have a selected number of bits of accuracy. Iterative multiplies are performed to iteratively increase the accuracy of the reciprocal approximation allowing a final quotient value of predetermined accuracy to be obtained.

[0006] The seed reciprocals are typically obtained from a ROM reciprocal look-up table, or equivalent PLA (programmed logic array). The number of table input index bits and table output bits of the seed reciprocals determines the size of the look-up table. More input bits allowing more bits of accuracy in the seed reciprocals reduces the necessary number of iterative multiply cycles, reducing division time, albeit at the cost of exponential growth in the reciprocal table size.

[0007] It will be appreciated that a floating point system or method that reduces the needed number of index bits and that reduces or eliminates the need for iterative cycles in resolving operations such as reciprocal, square root, and reciprocal square root would be useful..

Brief Description of the Drawings

[0008] The present disclosure may be better understood, and its numerous features and advantages made apparent to those skilled in the art by referencing the accompanying drawings.

[0009] FIG. 1 illustrates a block diagram of one embodiment of a processor in which an embodiment of the present disclosure may be implemented;

[0010] FIG. 2 illustrates a block diagram of a portion of a floating point unit (FPU) according to an embodiment of the present disclosure;

[0011] FIG. 3 illustrates a specific alignment of terms according to one embodiment of the disclosure; and

[0012] FIG. 4 illustrates a graph of the function $|2f-1|^3 \bullet 2^{-27}$ and the odd and terminal term error term accumulation used in establishing table values for an embodiment of the present disclosure.

[0013] The use of the same reference symbols in different drawings indicates similar or identical items.

DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

[0014] The present disclosure is directed to a method for evaluating arithmetic expressions, such as the square root, reciprocal square root, or reciprocal of a number, performed by a processor of a computer system. The processor evaluates the expressions using the quadratic equation

[0015] $Ax^2 + Bx + C$,

where the coefficient terms A, B, and C, are supplied by three coefficient lookup tables responsive to a common input as disclosed herein. The method results in high precision estimates without the use of iterative steps. In addition, the method taught herein uses compressed, or smaller, tables for the coefficient terms A, B, and C, thus minimizing the hardware requirements..

[0016] FIG. 1 illustrates a block diagram of one embodiment of a processor 100 in which an embodiment of the present disclosure may be implemented. While the present invention may be understood with reference to FIG. 1, this reference should not be construed in a limiting sense.

[0017] Processor 100 includes a bus interface unit 102 that controls the flow of data between processor 100 and the remainder of the data-processing system (not shown). Bus interface unit 102 is connected to both a data cache 104 and an instruction cache 106 for purposes of illustration.. Instruction cache 106 supplies instructions to branch and dispatch unit 108. Branch and dispatch unit 108 determines the sequence of instructions based on current data locations and the availability of load/store unit 112, fixed-point execution unit 114, and floating-point execution unit 116, and the nature of the instructions themselves. Branch and dispatch unit 108 issues the individual instructions to the appropriate logic unit (i.e. load/store unit 112, fixed-point execution unit 114, and floating-point execution unit 116), which in turn function together to implement request operations.

[0018] Those skilled in the art will appreciate that the details of either the construction of processor 100 or its operation may vary depending on the objectives influencing the design. For example, processor 100 may include register renaming multiple fixed-point execution units 114 for executing fixed-point instructions without data dependencies in parallel, or a memory management unit regulating the content of data cache 104 and a second level (L2) cache (not shown) outside processor 100.

[0019] In one embodiment, floating-point execution unit 116 comprises selecting three input operands (A, B, and C) from tables to allow for the implementation of floating point rounded monotonic quadratic functions based on a monotonic operation. For example, the tables can provide values to support providing a floating point rounded monotonic reciprocal function. FIG. 2 illustrates a specific pipelined architecture for implementing, accessing, and using values stored in such tables. Information regarding generation of the tables to be stored in FIG. 3 to support floating point rounded monotonic functions is described subsequent to FIG. 2.

[0020] FIG. 2 illustrates a block diagram of a portion of a high precision floating point unit (FPU) according to an embodiment of the present disclosure. The most significant bits of an input operand 202, the eight most significant bits for example, are used to index a seed read only memory (ROM) 204. It will be appreciated that operation information can also be used to index seed ROM 204. Operation information can include identification of various of operations, such as reciprocal or square root reciprocal. ROM 204 is part of a table look-up system that stores multiple component tables. For example, the ROM 204 can include tables to provide operands A, B, and C to be stored in registers 206, 208, and 210, respectively. The generation of the values contained in seed ROM 204 is discussed below.

[0021] One advantage of the implementation described herein is that the tables stored within ROM 204 can be accessed using a single index value having one-third the number of bits of the floating point result generated. For example, for a 24-bit result accurate to a Unit in the Last Place (ULP), only an eight-bit index is needed. The ability to provide this level of precision in the result using a ROM table of this size is advantageous.

[0022] Mid-range bits of input operand 202 are used to determine a nine-bit input to the square circuit 212, which may be a ROM or comprise a small multiplier, and a Booth recoding. Inverter 214 and multiplexer 216 are utilized to reduce the size of square circuit 212 by substantially one half, which is discussed in greater detail later in this disclosure. The operand output of square circuit 212 is stored in Booth recoded format in register 218 for input to a multiplier.

[0023] The least significant bits of input operand 202 are input to Booth recoder 220. The output of Booth recoder 220 is stored in register 222 for input to a multiplier.

[0024] Note that in the specific implementation illustrated, that the ROM table 204 is indexed by a portion of bits of the input operand that represent the operand's higher order bits. In addition, the ROM 204 is coupled to receive index bits that are mutually exclusive to the bits used to access the squaring circuit 212 and Booth recoder 220. However, the square circuit 212 and Booth recoder 220 are coupled to receive common, or overlapping, bits. While the portions of the input operand received at the ROM 212, ROM 204 and recoder 220 are illustrated as being adjacent bit sequences, they may be non-adjacent bit sequences in other embodiments.

[0025] Operand A from register 206 and the operand from register 218 are multiplied by multiplier 224. Multiplier 224 can be a carry save multiplier with redundant outputs and no ripple carry. Operand B from register 208 and the operand from register 222 are multiplied by multiplier 226. Multiplier 226 can also be a carry save multiplier with redundant outputs and no ripple carry. The redundant carry save outputs of multipliers 224 and 226 are added by four-to-two adder 228 producing a redundant carry save representation of the $Ax^2 + Bx$ term with the carry and sum values stored in registers 230 and 232, respectively. Operand C from register 210 is stored in register 234 to maintain a parallel pipeline. An implicit bit is appended to operand C from register 210 by appending implicit bit unit 236 forming operand C'. The carry and sum operands of the $Ax^2 +Bx$ term are added together with operand C' by three-to-one adder 238. The output of adder 238 is normalized into the format specified for the floating point output, and the resulting reciprocal value is stored in reciprocal register 242.

[0026] FIG. 3 is a block diagram illustrating an alignment of terms according to at least one embodiment of the present disclosure. During the table lookup and recoding stage of operation, the 8 most significant bits (79:72) of the input operand 203, excluding the implicit bit 80, are used as the index into the table of ROM 204. Bits 9 through 23 (71:57) of the input operand are used in calculation of the Bx term, and bits 9 through 18 (71:62) are used to access ROM 212 to calculate the x^2 component. A further step is used to permit the squaring circuit to be reduced in size. By centering the effect of the squared term, the size of the table used to calculate the square can be reduced by half by using what was formerly the most significant bit to invert the index into table. Since the square input is used as a multiplier input, the data that is stored in a ROM implementation of the squaring circuit can be precomputed recoded to the format expected by the multiplier, thus removing a Booth recoder from the path.. The x input to the Bx term is recoded in this stage.

[0027] In the multiply or second stage, both products are calculated in carry save multipliers. The results for both products are added together in a 4 to 2 adder, and the redundant result is passed to the add stage.

[0028] In stage three, or the add stage, an implicit bit is appended to the front of the C term. The two parts of the redundant number from the multiply stage are added with the C term to form the result. The values stored in the seed ROM are chosen to provide a rounded function as the result, when truncated to the output precision size during normalization . The rounded result is normalized in normalize block 240. For the reciprocal, square root reciprocal, and square root functions, the normalization comprisesforcing the output to an exact power of two if a carry out is detected from the adder stage. The normalize unit also removes the implicit bit before storing the result in the output register 242.

[0029] The tables stored in ROM 204 are chosen to allow for implementing a floating point rounded monotonic quadratic function based on a monotonic operation. Specifics of generating the tables of ROM 204 are described below.

TABLE COMPRESSION

[0030] A monotonic operation applied to a single operand is a monotonic function of that operand. This is the case for the monotonic operations reciprocal, square root, and square root reciprocal. A monotonic function, as the term monotonic implies, is always changing in the same direction. A monotonic increasing function of a variable y increases or stays constant as y increases, but never decreases. A monotonic decreasing function of y decreases or stays constant as y increases, but never increases. Each term in a monotonic series is, in an increasing monotonic, greater than or equal to the one before it. Or, in a decreasing monotonic, less than or equal to the one before it.

[0031] A floating point rounded function over an interval has a discrete set of floating point inputs over that interval. The value of the floating point rounded function over the interval is a step function, with the steps decreasing or remaining the same as the floating point inputs increase for a decreasing monotonic floating point rounded function, and with the steps increasing or remaining the same as the floating point inputs increase for an increasing monotonic function.

[0032] It is sufficient for the reciprocal operation to consider floating point inputs over the interval [1, 2). Thus for example $y = 1.b_1 b_2 \ldots b_{23}$ provides the 2^{23} (about 8 million) input values for the 24 bit precision specified in the IEEE standard single precision format. For the square root and square root reciprocal operations it is necessary to consider two input binades covering [1, 4).

[0033] For evaluation of a floating point rounded polynomial function, such as the quadratic $A y^2 + B y + C$, where all coefficients A,B,C are determined from a leading bit portion such as $y_8 = 1.\ b_1 b_2 \ldots b_8$ of y, the floating point rounded quadratic function is then a piecewise quadratic function, here having $2^8 = 256$ pieces over [1, 2).

[0034] The verification that such a floating point rounded quadratic function is monotonic can be verified by observing that each piece is monotonic and by verifying that the joining points of the pieces (here 255 in number) preserver monotonicity. This test confirming monotonicity as well as tests for the whole single precision range of 8–16 million cases for each reciprocal, reciprocal square root and square root reciprocal are easy to perform in addition to theoretical arguments from the level of accuracy and the derivatives of the function.

[0035] The simplicity of these tests allows perturbation of the coefficients to improve the percent round to nearest results or to reduce table size without losing the monotonicity or unit in the last place accuracy of the floating point rounded function realizing the monotonic operation.

[0036] The following paragraphs describe the calculations to provide an 8-bit *y* table solution for table compression such that a plurality of tables, three in our example, are indexed by an index value three times smaller bitwise than the floating point result, i.e., 8-bit tables produce a 24 bit floating point result.

<u>TABLE COMPRESSION (8-bit *y* table solution)</u>

[0037] Let

$$y = \underbrace{1.b_1b_2..b_8}_{y_8}\ \underbrace{b_9b_{10}..b_{23}}_{f} = y_8 + f2^{-8}\text{, where} \qquad (1)$$

$$y_8 = 1.b_1b_2..b_8\text{, and} \qquad (2)$$

$$f = 0.b_9b_{10}..b_{23}\ \text{(15 bits), where } 0 \le f \le 1 - 2^{-15}. \qquad (3)$$

The midpoint is then given by

$$y_8 + 2^{-9} = 1.b_1b_2..b_8, \qquad (4)$$

and the center point by

$$(2f-1) = (b_9b_{10}...b_{23}) - 1\text{, where } -1 \le (2f\text{-}1) \le 1 - 2^{-14}. \qquad (5)$$

Thus the expression for y is given by

$$y = (y_8 + 2^{-9}) + (2f-1)2^{-9}. \qquad (6)$$

The following identity for the reciprocal of *y* is given by the expression:

$$\frac{1}{y}=\frac{1}{y_8+2^{-9}}-\frac{(2f-1)}{y_8+2^{-9}}\frac{1}{y}2^{-9} \qquad (7)$$

[0038] The preceding equation (7) may be checked by placing terms over a common denominator.

[0039] Iterative substitution for 1/y two times yields a cubic expression in (2*f*-1) as follows:

$$\frac{1}{y}=\underbrace{\frac{1}{y_8+2^{-9}}}_{A}\underbrace{-\frac{(2f-1)}{(y_8+2^{-9})^2}2^{-9}}_{-B2^{-9}}\underbrace{+\frac{(2f-1)^2}{(y_8+2^{-9})^3}2^{-18}}_{+C2^{-18}}\underbrace{-\frac{(2f-1)^3}{(y_8+2^{-9})^3}\left(\frac{1}{y}\right)2^{-27}}_{-D2^{-27}} \qquad (8)$$

[0040] Note the algebraic expression 8 is an exact equation for the reciprocal, where the first three terms can be evaluated to as much accuracy as desired by a lookup table indexed by 8-bits along with appropriate multiplications. Only the fourth term, which constitutes less than 1/8 of a unit relative to our target precision of 24-bits, needs to be approximated.

[0041] Tables for the A, B, and C coefficients for a reciprocal and a square root reciprocal which have been generated utilizing the mathematical methodology herein are presented in the Appendix.

[0042] Our invention employs several techniques to generate the coefficients C_0, C_1, C_2 of the rounded floating point function *approx (1/y)* to reduce table size and computation effort while maintaining the monotonicity property with unit in the last place (ulp) accuracy in realizing the reciprocal operation.

- The coefficient C_0 includes the half ulp increment so that the final rounding occurs simply by truncating the result during normalization to the floating point output format.

- The coefficient C_2 includes compensation for the rounding error in providing C_0, so that the total coefficient rounding errors from C_0 and C_2 are correlated to provide a net rounding error no worse than the worse case of a single rounding error, as described in co pending application 10/108,251, filed on March 26, 2002, entitled APPARATUS AND METHOD FOR MINIMIZING ACCUMULATED ROUNDING ERRORS IN COEFFICIENT VALUES IN A LOOKUP TABLE FOR INTERPOLATING POLYNOMIALS, naming inventor David W. Matula, which is incorporated herein by reference.
- The coefficient of the linear term C_1 includes a linear approximation to the cubic term of equation 8 to increase the accuracy so as to reduce the table size for the final 3 coefficients C_0, C_1, C_2 of our floating point function *approx (1/y)*.

ERROR ROUNDOFF METHODOLOGY

[0043] If we let

$$C_1 = RN_{18}\left(\frac{1}{(y_8+2^{-9})^2} + \frac{169/225}{(y_8+2^{-9})^3 y} 2^{-18} \right)$$

and

$$\rho_1 2^{-19} = \left(\frac{1}{(y_8+2^{-9})^2} + \frac{169/225}{(y_8+2^{-9})^3 y} 2^{-18} \right) - C_1$$

Then.

$$\frac{1}{y} = C_0 + C_2 Q 2^{-18} + \rho_a 2^{-28} + C_2 \rho_Q 2^{-28} + C_2(2f-1)\rho(f) 2^{-27} + C_1(2f-1)\rho(f) 2^{-9} + (2f-1)\rho_1 2^{-28} + \left(\frac{\bar{f}}{\bar{y}^4}\right) 2^{-29} + \partial' 2^{-28}$$

where

$$\bar{f} = 32\left(f - \frac{1}{2}\right)\left(f - \frac{1}{15}\right)\left(f - \frac{14}{15}\right)$$, and ∂' is less than 1.

$$\bar{y}^4 = (y_8 + 2^{-9})^3 y.$$

[0044] Let us define an approximation, $approx\left(\frac{1}{y}\right)$, in terms of our table lookup and multiply accumulate in finite precision by the expression:

$$approx\left(\frac{1}{y}\right) = C_0 + C_1(2f-1)2^{-9} + C_2Q2^{-18}$$

Then the error term identification is as follows:

$$\left(\frac{1}{y}\right) - approx\left(\frac{1}{y}\right) = \rho_a 2^{-28} + (2f-1)\rho_1 2^{-28} - \left(\frac{\bar{f}}{\bar{y}^4}\right)2^{-29} + C_2\rho_Q 2^{-28} + C_2(2f-1)\rho\big(f\big)2^{-27}$$

[0045] Where the portion of the expression $\rho_a 2^{-28}$ represents the even terms roundoff, the portion $(2f-1)\rho_1 2^{-28} - \left(\frac{\bar{f}}{\bar{y}^4}\right)2^{-29}$ represents the odd terms roundoff and function termination, the portion $C_2\rho_Q 2^{-28}$ represents the squaring term roundoff, and the portion of the expression $C_2(2f-1)\rho\big(f\big)2^{-27}$ represents the approximation error for the square.

[0046] The behavior of the portions of the error terms expression can be understood with reference to Table 1.

Table 1 – Behavior of Error Terms

$\rho_a 2^{-28}$	$(2f-1)\rho_1 2^{-28} - \left(\frac{\bar{f}}{\bar{y}^4}\right)2^{-29}$	$C_2\rho_Q 2^{-28}$	$C_2(2f-1)\rho\big(f\big)2^{-27}$
Essentially uniform over $(-1,1) \bullet 2^{-28}$, independent of y and f.	Sum can be correlated to be bounded by $\frac{5}{4}2^{-28}$.. Non linear term diminishes rapidly with y→2.	Essentially $\frac{\rho_Q}{y^3}2^{-28}$. Damps rapidly with y→2.	Essentially $\frac{2f-1}{y^3}\rho(f)2^{-27}$. Damps rapidly with y→2 and $\lvert 2f-1\rvert$ →0.

[0047] The damping of error terms can be understood with reference to Table 2.

Table 2 – Error Term Damping

Error Term	Unrestricted Upper Bound	$y \geq 1\frac{1}{4}$ Upper Bound	$y \geq 1\frac{1}{2}$ Upper Bound	$\lvert 2f-1 \rvert \leq \frac{1}{2}$ Upper Bound
$\rho_\alpha 2^{-28}$	1	1	1	1
$(2f-1)\rho_1 2^{-28} - \left(\frac{\bar{f}}{\bar{y}^4}\right)2^{-29}$	3/2	1.2	1.1	1
$C_2 \rho_Q 2^{-28}$	1	0.5	0.3	1
$C_2(2f-1)\rho(f)2^{-27}$	2	1	0.6	1
Independent Summation Bound (units)	5.5 (x 2^{-28})	3.7 (x 2^{-28})	3 (x 2^{-28})	4 (x 2^{-28})

[0048] It is preferable to have $\left|\left(\frac{1}{y}\right) - approx\left(\frac{1}{y}\right)\right| < 3.5 \bullet 2^{-28}$. The difficult cases should be found in the restricted range $1 \leq y \leq \frac{5}{4}$, and $\lvert 2f-1 \rvert > \frac{1}{2}$. The worst case compounding of all four errors above is unlikely given the relatively small number of cases. It should be noted that there are only about 64 values for ρ_α and ρ_1 over the range $1 \leq y \leq 1\frac{1}{4}$, and only about 256 values for ρ_Q over $\lvert 2f-1 \rvert \geq \frac{1}{2}$. Perturbation of coefficients by one, at most, unit in the last place (ULP) each can also avoid worst cases.

[0049] An advantage of the present disclosure for the use of floating point rounding to realize a monotonic operation to construct the coefficient tables shown in Appendix A is that the higher order (cubic) term approximation can be incorporated into the linear term such that no fourth coefficient term table is needed to realize a higher than quadratic level approximation. This concept is illustrated in FIG. 4, which illustrates a graph of the function $|2f-1|^3\ 2^{-27}$ and the odd and terminal term error term accumulation in which the present disclosure may be implemented. The functions graphed are sign symmetric about f=1/2.

[0050] The above-disclosed subject matter is to be considered illustrative, and not restrictive, and the appended claims are intended to cover all such modifications, enhancements, and other embodiments that fall within the true spirit and scope of the present invention. Thus, to the maximum extent allowed by law, the scope of the present invention is to be determined by the broadest permissible interpretation of the following claims and their equivalents, and shall not be restricted or limited by the foregoing detailed description.